DENBench™1.0

software benchmark data book

www.eembc.org

An Industry-Standard Benchmark Consortium

Table of Contents

AES 2

DES 4

Calculating the DENmark™ and other DENBench™ Consolidated Scores 6

High-Pass Grey-Scale Filter 9

Huffman Decoding 14

MP3 Decode 15

MPEG-2 Decode 18

MPEG-2 Encode 22

MPEG-4 Decode 28

MPEG-4 Encode 34

RGB to CMYK Conversion 41

RGB to YIQ Conversion 46

RSA 51

DENBench™ Version 1.0

Benchmark Name: AES

Highlights

- **Benchmarks the Rijndael Algorithm of the Advanced Encryption Standard (AES) algorithm, named for creators Joan Daemen and Vincent Rijmen**
- **Roundtrip implementation plus integration of the FIPS test assures accuracy**
- **A component of the EEMBC Cryptography sub-suite**
- **AES tends to replace DES and Triple-DES as the preferred encryption algorithm for maximum-security applications in governmental, HDTV, satellite, and vital data-security applications**

Application and Restrictions

The Advanced Encryption Stand (AES) benchmark provides an indication of the potential performance of a microprocessor or Digital Signal Processor (DSP) subsystem doing AES cryptographic encryptions and decryptions. The AES cipher is used in numerous cryptographic protocols, including Transport Layer Security (TLS), Secure Socket Layer, (SSL), Secure Shell, (SSH), and Internet Protocol Security (IPSEC). AES is a royalty-free Federal Information Processing Standards (FIPS) approved standard intended to ultimately replace Digital Encryption Standard (DES). The EEMBC benchmark and its source code are subject to the following restrictions:

> *Joan Daemen and Vincent Rijmen, the developers of the Rijndael algorithm, submitted this software package to the National Institute of Standards and Technology (NIST) during the Advanced Encryption Standard (AES) development effort. This software is distributed in compliance with export regulations (see below), and it is intended for non-commercial use only. NIST does not support this software and does not provide any guarantees or warranties as to its performance, fitness for any particular application, or validation under the Cryptographic Module Validation Program (CMVP) http://csrc.nist.gov/cryptval. NIST does not accept any liability associated with its use or misuse. This software is provided as-is. By accepting this software the user agrees to the terms stated herein.*

The EEMBC AES Benchmark Software is subject to the following Export Restrictions (exportation from the United States of America to non-USA countries): *Implementations of cryptography are subject to United States Federal Government export controls. Export controls on commercial encryption products are administered by the Bureau of Export Administration (BXA) http://www.bxa.doc.gov/Encryption/ in the U.S. Department of Commerce. Regulations governing exports of encryption are found in the Export Administration Regulations (EAR), 15 C.F.R. Parts 730-774. Compliance with export restrictions is the responsibility of each individual EEMBC member, not EEMBC, Inc. itself.*

Benchmark Description

Rijndael is an iterated block cipher with a variable block length and a variable key length. The block length and the key length can be independently specified to 128, 192, or 256 bits. The EEMBC AES benchmark implements all three of these key lengths for each iteration. Like most cryptographic functions, there is an array (or "block") that is subjected to multiple transformations. This benchmark performs 1000 Monte Carlo Tests (MCT) for 16 passes, and does a complete round-trip encryption, followed by decryption, to verify correctness. Implementing the FIPS tests inside the code itself further enhances correctness. The code faithfully implements the Wide Trail Strategy to deflect against layer attacks, and implements all three layers: linear-mixing, non-linear layer, and key addition layer.

Unlike the other EEMBC cryptography benchmarks (which were created based on the SSLEAY 0.9 project), the EEMBC AES benchmark was created from the baseline source code and specification by Vincent Rijmen and Joan Daemon found at http://csrc.nist.gov/CryptoToolkit/aes/. It supports the FIPS test found at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

The data that is supplied is proprietary to EEMBC. In addition, EEMBC has its own, secret data that is used to double-check accuracy of implementation.

Analysis of Computing Resources

The benchmark is computationally challenging: addition, multiplication, extensive use of division, bit shifting, matrix math, bitwise operators such as XOR, and other operators are used. It is implemented in integer math. This benchmark is almost exclusively CPU bound, and the quality of the math library as well as memory library has an effect on performance. Memory moves are performed repeatedly, so optimized C library `mem*` functions would improve performance, but without overwhelming the basic math computations. Sophisticated superscalar architectures scheduled by sophisticated compilers (or assembly language implementations) can take advantage of some parallelism.

DENBench™ Version 1.0 Benchmark Name: DES

Highlights

- **Benchmarks the Digital Encryption Standard (DES) algorithm**
- **Created in part from SSLEAY sources, the open source Netscape Secure Socket Layer source code base**
- **Roundtrip implementation and self-checking assures accuracy**
- **A component of the EEMBC Cryptography Sub-suite**
- **DES is a popular cryptographic algorithm (especially in Triple-DES configuration) used widely in eCommerce applications, including mobile phones m-commerce**

History, Applications and Restrictions

The Digital Encryption Standard (DES) benchmark is a cipher algorithm that provides an indication of the potential performance of a microprocessor or digital signal processor (DSP) subsystem doing DES cryptographic encryptions and decryptions. The DES cipher is used in numerous cryptographic protocols, including transport layer security (TLS), secure socket layer, (SSL), secure shell (SSH), and Internet protocol security (IPSEC). A history of the cipher's development is available at http://en.wikipedia.org/wiki/DES.

Although it is vulnerable to certain, extremely computationally intensive cracking attacks (on the order of a successful crack in 24 hours), DES remains popular in many eCommerce (internet) and m-commerce (mobile) applications but has been replaced for the most secure applications by the Advanced Encryption Standard (AES), which is also part of the EEMBC benchmark suite.

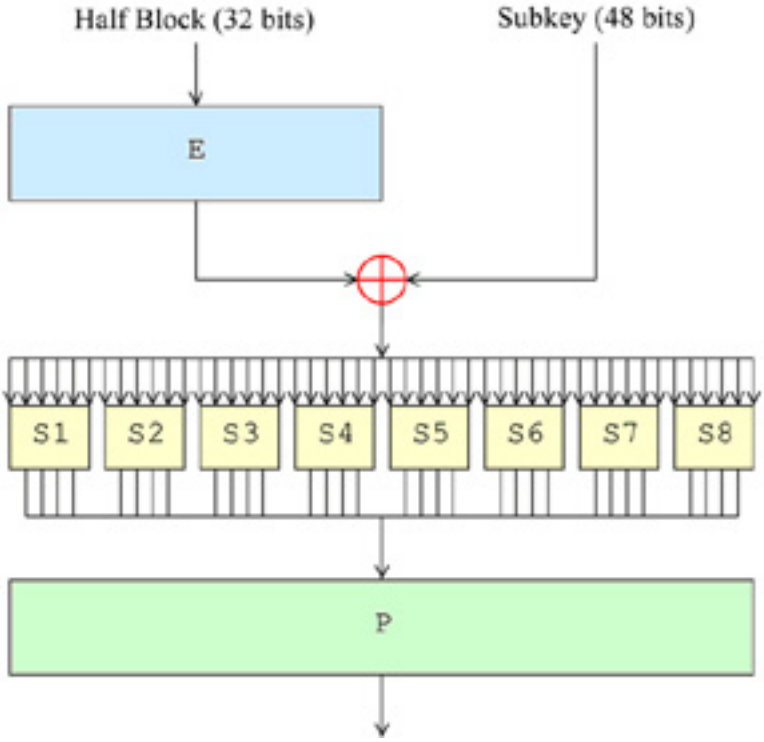
This benchmark, and the source code, is subject to the following restrictions:

This software is subject to export restrictions from the United States of America to non-USA countries. Export and re-export controls on commercial encryption products are administered by the Bureau of Industry and Security (BIS) of the U.S. Department of Commerce. Rules governing exports and re-exports of encryption items are found in the Export Administration Regulations (EAR), 15 C.F.R. Parts 730-774. Sections 740.13, 740.17 and 742.15 of the EAR are the principal references for the export and re-export of encryption items. Further information is available from http://www.bis.doc.gov/Encryption.

Benchmark Description

DES is an iterated block cipher with a fixed block length of 64 bits and a fixed key length of 64 bits. However, only the first 56 bits are used; the other 8 bits are for parity checking (hence DES is considered 56-bit encryption). EEMBC implements the correct key length. Like most cryptographic functions, there is an array (or "block") that is subjected to multiple transformations (in

the case of DES, 16). Unlike the AES benchmark implementation [correct?], EEMBC does not implement the FIPS tests on DES. Checking is by cyclical redundancy checksum (CRC).



The input data for the DES benchmark is proprietary to EEMBC but [what characteristics?].

Analysis of Computing Resources

The benchmark is computationally challenging: addition, multiplication, extensive use of division, bit shifting, matrix math, bitwise operators such as XOR, and other operators are used. It is implemented in integer math. This benchmark is almost exclusively CPU bound, and the quality of the math library as well as memory library has an effect on performance. Memory moves are repeatedly performed, so optimized C library mem* functions would improve performance without overwhelming the basic math computations. Superscalar and VLIW architectures scheduled by sophisticated compilers (or assembly language implementations) can take advantage of some parallelism. Many of the computationally challenging functions can be offloaded to hardware acceleration logic.

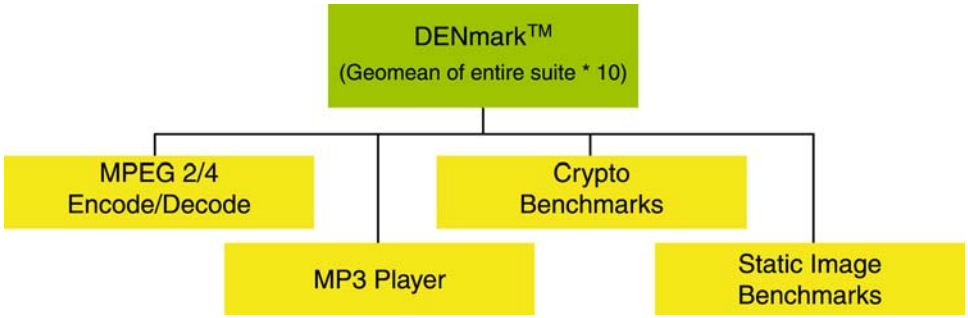
DENBench™ Version 1.0

Benchmark Name: Calculating the DENmark™ and other DENBench™ Consolidated Scores

The DENmark,™ MPEG Decodemark™, MPEG Encodemark™, Cryptomark™, and Imagemark™ are single-number scores that EEMBC provides, in addition to scores based on individual benchmark applications within its DENbench suite, to enhance the presentation of comparative data on processor performance. These numbers are intended to provide a first-order representation of processor performance in tasks related to digital entertainment applications. The detailed scores on individual benchmarks and datasets will continue to offer the highest value to system designers, allowing comparison of the individual applications that are specific to their designs.

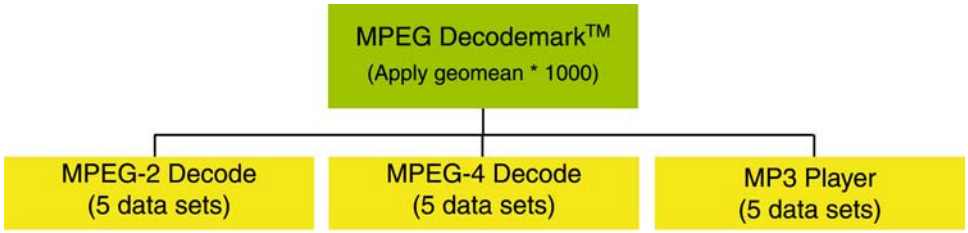
DENmark™

An overall "DENmark" score provides a single-number performance rating for the DENbench suite. A member must run all tests in this suite, except for the MPEG-2 Encode (Floating-Point) benchmark with its five datasets, in order to derive a DENmark score.



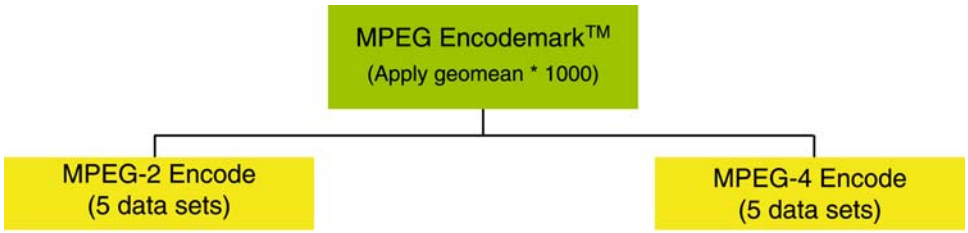
MPEG Decodemark™

The MPEG Decodemark consolidated scores provide a snapshot of performance in specific test groups:



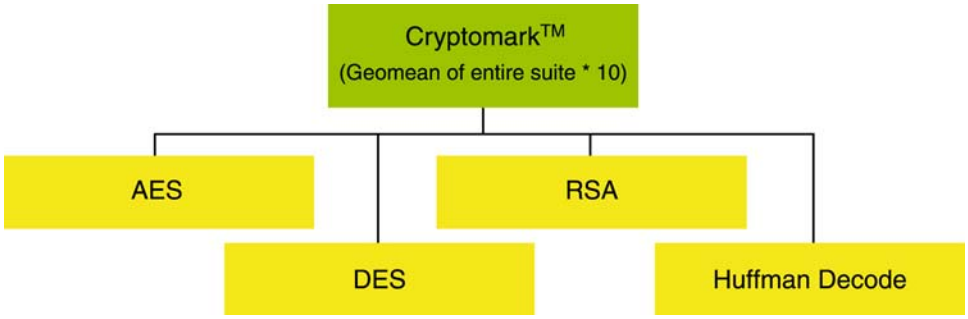
MPEG Encodemark™

The MPEG Encodemark consolidated scores provide a snapshot of performance in specific test groups:



Cryptomark™

The Cryptomark consolidated scores provide a snapshot of performance in specific test groups:



Members are not required to publish all scores in the DENbench suite. In other words, they can choose to run only the benchmark applications that make sense for their processors.

The method used in calculating each of the "mark" scores is shown in the following table. The geometric mean of each 'mark' is multiplied by a normalizing factor intended to keep most of the 'marks' within the same order of magnitude.

Consolidated score name	Apply geometric mean to:	The *n*th root used to calculate geometric mean	Multiply geometric mean result by this normalizing factor:
MPEG Decodemark	the 5 scores from each of MPEG-2 Decode benchmark, MPEG-4 Decode benchmark, MP3 player benchmark	15	1000
MPEG Encodemark	the 5 scores from each of MPEG-2 Encode benchmark and MPEG-4 Encode benchmark	10	1000

Consolidated score name	Apply geometric mean to:	The *n*th root used to calculate geometric mean	Multiply geometric mean result by this normalizing factor:
Cryptomark	each of the scores from AES, DES, RSA, and Huffman Decode benchmarks	4	10
Imagemark	the 7 scores from each of RGB/YIQ, RGB/HPG, RGB/CMYK, JPEG Compression, and JPEG Decompression	35	10
DENmark	each of the 64 individual scores/datasets from the DENbench suite	64	10

The DENbench suite also includes an MPEG-2 Encode (Floating-Point) benchmark with five datasets. The consolidated score is calculated by taking the geometric mean of the five scores generated and multiplying the result by 100. The result does not contribute to the overall DENmark score and no distinct "mark" score is calculated.

Note on the use of geometric mean versus arithmetic mean

The geometric mean is used in calculating the consolidated, single-number scores to assure equal weighting for all benchmarks in each category. An arithmetic mean of raw data is not statistically valid because of the extremely wide variation of the results. Individual results that yield a very small number of iterations would have virtually no effect on an arithmetic mean when combined with raw results that yield a very high number of iterations. In effect, an arithmetic mean of results would impose an arbitrary weighting system that heavily favors the tests with the most iterations per second.

DENBench™ Version 1.0

Benchmark Name: High-Pass Grey-Scale Filter

Highlights

- **Benchmarks performance for image processing used in digital still camera and other digital imaging products**
- **Explores 2-D data array access and multiply/accumulate capability**
- **Integer implementation**
- **Seven datasets provide a larger workload compared to the single dataset of ConsumerBench Version 1.1**
- **Inputs are black and white Portable Graymap (.pgm) files**
- **Implements Non-Intrusive Cyclical Redundancy Checksum (CRC) to Check Output Quality**

Application

A high pass gray-scale filter is used in the front end processing of digital still cameras (DSCs). RGB data from either CCD or CMOS sensors is pre-processed by this filter to deliver image enhancement, and then passed to the JPEG image compression processing. This filter takes a blurry image and sharpens it with a 2-dimensional spatial filter. DSCs implement this filter either in software or hardware, with software giving the flexibility to add customization for picture quality. The number of filter taps can vary from 3(H) x 3(V) to more than 5(H) x 5(V). This benchmark is one of the most frequently used algorithms in image processing and represents a good measure of the CPU performance in digital imaging products.

Benchmark Description

This benchmark explores the target CPU's capability to perform two-dimensional data array access and multiply/accumulate calculations. For each pixel in the image, the filter calculates the output result from the 9 pixels (including the center pixel) multiplied by filter coefficients, accumulated and then shifted left by 8 bits.

The two-dimensional coefficients used here are:

$$\begin{bmatrix} F11 & F21 & F31 \\ F12 & F22 & F32 \\ F13 & F23 & F33 \end{bmatrix} = \begin{bmatrix} -28 & -28 & -28 \\ -28 & 255 & -28 \\ -28 & -28 & -28 \end{bmatrix}$$

Each pixel is computed according to the following equation:

```
PelValue = (Short)(  F11*P(c-w-1) +F21*P(c-w)  +F31*P(c-w+1)
                     +F12*P(c-1)    +F22*P(c)     +F32*P(c+1)
                     +F13*P(c+w-1)+F23*P(c+w) +F33*P(c+w+1) )

Out = (Byte)(PelValue >>8);
```

Here, P(i) is the pixel intensity, c is the center location of the filter window, w is the width of the input image. The data type of P(i) is byte, and the two-dimensional data is arranged in a linear way. Therefore addition or subtraction of the horizontal image width w and offset of -1 or +1 are required to retrieve the two-dimensional window data. The accumulation is performed as 16-bit data and the final output data is converted to byte data after a shift right by 8 bits. The top/left and right/left borders are blacked out by assigning BLACK a value of 0.

The input data sizes vary and use monochrome .pgm files, performing monochrome or gray-scale calculation. It is not an RGB calculation where the same process is performed three times. Usually the enhancement is performed just in the luminance signal Y, which is the gray-scale signal. If the benchmark score is extrapolated for a larger image, the processing time will be almost linearly proportional to the pixel count (e.g. for a 640 x 480 image, it will be multiplied by 4). The iteration/sec score will be the inverse (e.g. for a 640 x 480 image, iterations/seconds will be multiplied by .25).

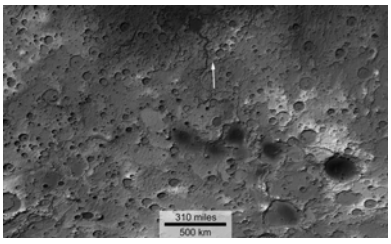
Description of Datasets

Rose Small

The dimensions are 227x149, 256 colors, 146 colors used in pgm format.

Goose

The dimensions are 320x240, 256 colors. The image has 254 unique colors in .pgm format.



Mars Former Lakes

Mars Former Lakes is a NASA graphics picture. The dimensions are 800x482, 16 million colors. The image has 255 unique colors in .pgm mode.

Dragon Fly

Dragon Fly is an image containing highlights and a wide range of contrast. The dimensions are 606x896, 16 million colors. The image has 162,331 unique colors in .pgm format.

EEMBC Group Shot

EEMBC Group Shot is a snapshot of attendees at an EEMBC Board of Directors meeting in 2003. The dimensions are 640x480, 16 million colors. The image has 253 colors in this .pgm file.

David and Dogs

David and Dogs is a snapshot of David Weiss and his dogs Sandy, Toga, and Trudy during a rare snowstorm in Austin. It is used as a grayscale image, with good contrast details in the melting snow. The dimensions are 564x230, 256 shades of gray. The image has 215 unique colors.

Mandrake

Mandrake is a close up picture of a Mandrill Baboon (sometimes misnamed as "Mandrake"). It has a lot of detail and colors. It has been the default image for the filter benchmarks in both color and gray scale. The dimensions are 320x240, 16 million colors. The image has 213 unique colors in this .pgm version.

Galileo

Galileo is a NASA composite image based on actual images of the Jupiter and several of its moons. The dimensions are 290x415, 16 million colors. The image has 36,557 unique colors, and also contains "real black" for over 30% of the picture, which is interesting from an optimization perspective. This .pgm file has 256 colors.

Output quality is measured using Non Intrusive CRC code developed by the EEMBC Certification Laboratory (ECL, LLC). It does not affect the benchmark score.

Analysis of Computing Resources

Out of the Box Benchmark: A "for loop" calculates the filter output one pixel at a time. For one pixel calculation, the center pixel itself and the eight neighbor pixel data should be loaded. This is a time consuming process, considering the offset/width index calculation, and the time spent for the memory or cache access. Higher performance would be expected from a microprocessor with a single-cycle MAC unit.

Full-Fury Benchmark: Because of the simple structure of the multiplication and accumulation, a VLIW or SIMD architecture with multiple MAC units is able to offer a simple acceleration. Another possible optimization is loading multiple bytes at a time, although a SIMD architecture may show some overhead for the rearranging the data to feed the SIMD engine. Regarding the memory architecture, the image data is repeatedly used for the consecutive window and can benefit from a data cache. The code size is trivial and will easily fit in to a small L1 instruction cache.

DENBench™ Version 1.0

Benchmark Name: Huffman Decoding

Highlights

- **Benchmarks potential performance of a processor in a digital camera and is modeled on picture data (YUV data)**
- **Huffman Decoding is a key algorithm in JPEG, MPEG, and compression schemes**
- **Integer implementation**
- **Stresses table lookup, bit manipulation, shifting**
- **Implements CRC to Check Output Quality**

Application	Huffman decoding is a key algorithm in JPEG, MPEG, and other compression schemes used in digital cameras. Details on Huffman coding and decoding are available here: http://en.wikipedia.org/wiki/Huffman_coding
Benchmark Description	The Huffman Decoding benchmark initializes the AC and DC chrominance and luminance tables (4 tables), fills the various buffers, and then performs the Huffman decoding function using a fairly standard Huffman implementation.
Analysis of Computing Resources	This benchmark concentrates mostly on lookup, bit manipulation, and shifting rather than file I/O. Memory-to-memory operations are important for performance, as intermediate values are constantly being stored.
Optimizations Allowed	**Out of the Box / Standard C** **Full Fury / Optimized**

- The C code must not be changed for Out of the Box unless it must be modified to get it to compile. All changes must be documented and must not have a performance impact.
- For Out of the Box, additional hardware can be used if it does not require code changes.
- All optimized libraries must be part of the standard compiler package, and/or available to all customers
- The EEMBC Test Harness Regular or Test Harness Lite may be used. Test harness changes may be made for portability reasons if they do not impact performance
- For Optimized, the basic algorithm may be changed and/or the code can be rewritten in assembler. We report PSNR scores to help you judge quality of computational processing.
- For Optimized, optimized libraries can be used if they are publicly available.
- For Optimized, hardware-assist can be used if it is on the same processor as that being benchmarked.
- For Optimized, in-lining is allowed.
- Additional data files may be used by ECL during certification to ensure the correctness of the optimized benchmark. You should not assume data patterns during optimization.

DENBench™ Version 1.0

Benchmark Name: MP3 Decode

Highlights

- **Benchmarks potential performance of an MP3 player's processor subsystem**
- **Uses five different test files**
- **Integer implementation derived from the MSSG ISO sources**
- **Implements PSNR to check the output quality**

Application

The MP3 Decoder benchmark provides an indication of the potential performance of a microprocessor subsystem running an MP3 player application.

Benchmark Description

The benchmark is an integer implementation of the ISO 13818-3 MPEG-2 Layer 3 decoder with lower sampling frequency extension. Normal sampling frequencies are 32 kHz, 44.1 kHz (typical CD-ROM audio), or 48 KHz. Lower sampling frequencies are 16 KHz, 22.05 KHz, or 24 KHz. We selected a lower sampling to reflect that which is often used in PDAs, mobile phones, and on websites where bandwidth is a concern. The benchmark does not include the standard MPEG optimizations, i.e. neither the 0.9.3 nor the 0.9.5 optimizations are implemented because we selected pure "reference code" as the baseline for this benchmark.

The benchmark includes both Huffman decoding and modified inverse discrete cosine transform (iDCT) routines.

The benchmark simulates the decoding and playback by encapsulating data statically (rather than through file I/O) of the following MP3 encoded files:

- JUPITER.mp3: A faithful rendition of "Jupiter, the Bringer of Jollity" from Gustav Holst's *The Planets*. Encoded at 160 KBps, stereo. Dynamics suggest a full range of signals.
- music128stereo.mp3: A sophisticated set of music and noise samples spanning the full dynamic range. This one is encoded at 128 KBps (constant), stereo, at very high quality, and consists of one minute's worth of playback. This file is about 993KB on disk (based on Windows XP NTFS filesystem). 128 KBps is considered the best MP3 rate for quality in portable players.
- music48_128stereo.mp3: Same set of music as above, but encoded in stereo with a variable bit rate of between 48 KBps and 128 KBps. Very high quality. 855 KB on disk suggests that most of it was encoded at high bit rates by our encoder at the EEMBC Certification Laboratory (ECL), but some at lower bit rates. 48 KBps is often the maximum bit rate used for cellular/mobile phone.
- music64stereo.mp3: Same set of music as above, but encoded at 64 KBps, an ideal compromise on quality vs. size. Constant rate, stereo. 470 KB on disk.
- music48mono.mp3: Same set of music as above, but encoded in monaural (mono), not stereo, at 48 KBps constant. 353 KB on disk.

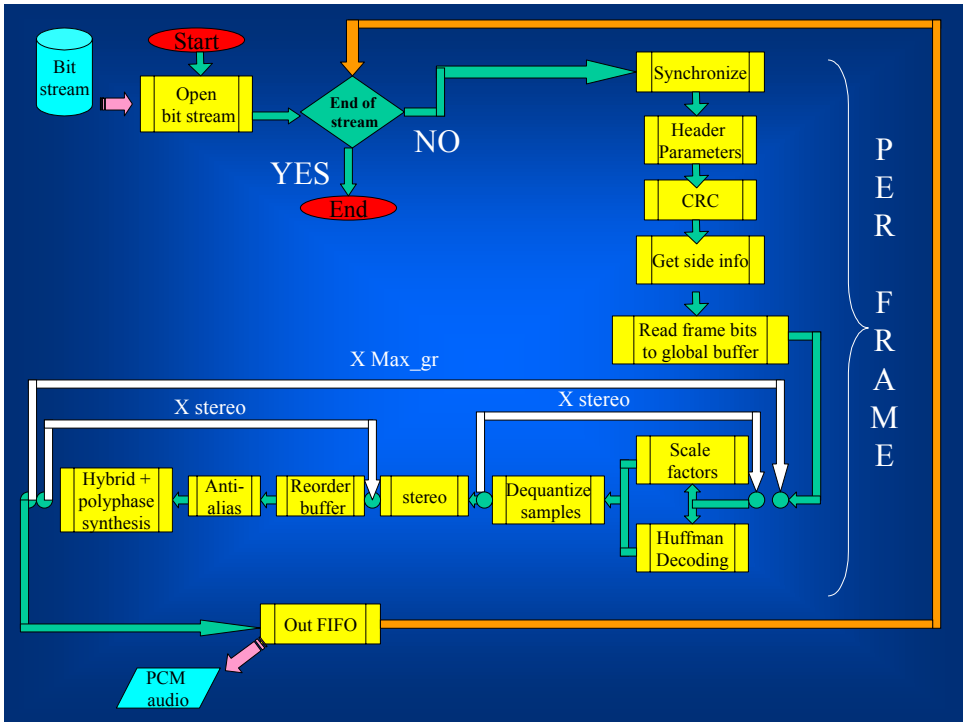
Processing consists of:

1. Reading the selected MP3 file.
2. Reading and interpreting the header information.
3. Read and decode frames of data.
4. Process the data based on the header information.
5. Output music as two channels of 16-bit pulse code modulated data. This data is placed into an AIFF format file which is supported by a wide range of players.
6. A PSNR value is calculated for each PCM frame against a reference AIFF file. These frame scores are aggregated into a single PSNR score for the benchmark.

A single iteration of the benchmark is complete when the end of the input file is reached and no more data is available to be processed.

A way to measure the quality of the output based on peak signal-to-noise ratio (PSNR) code was developed by ECL and implemented in accordance with the Consumer Subcommittee, Technical Advisory Board (TechTAG), and EEMBC Board of Directors. PSNR is a decibel measurement of noise power. PSNR is consistent for the industry and widely used to measure picture quality and audio quality. PSNR is measured outside the benchmark timing loop, on the host side processor (not the embedded target).

The flow diagram is as follows:



Analysis of Computing Resources

This is an integer-only implementation of MPEG-1/2 Layer 3 audio and is a benchmark that concentrates mostly on computational processing rather than file I/O. In the following order, synchronization and error checking, Huffman decoding, re-quantization (using inverse discrete cosine transform, iDCT), and reordering are performed.

An Industry-Standard Benchmark Consortium

DENBench™ Version 1.0 MPEG-2 Decode

Highlights

- **Benchmarks potential performance of an MPEG-2 Decoder**
- **Five different test files stress different decoder aspects**
- **Integer implementation**
- **Implements PSNR to check output quality**
- **Based on the ISO reference source**

Application

The MPEG-2 Decoder benchmark provides an indication of the potential performance of a microprocessor subsystem running an MPEG-2 Decoder application, such as those found in a DVD player or a digital set-top box.

Benchmark Description

The benchmark contains a fixed point (integer) implementation of the MSSG ISO sources. The MPEG-2 Decoder uses a standard reference implementation of the core algorithm, including Huffman decoding and modified Inverse Discrete Cosine Transform (iDCT) routines.

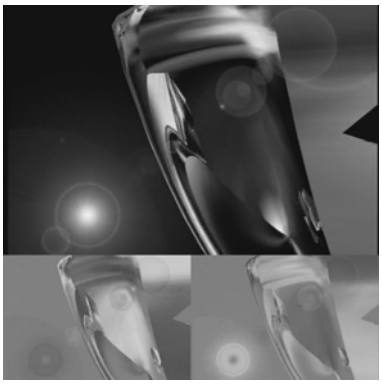
The fixed-point implementation base algorithms of `fdctint.c` and `jfdctint.c` are based on C. Loeffler, A. Ligtenberg and G. Moschytz, "Practical Fast 1-D DCT Algorithms with 11 Multiplications," Proc. Int'l. Conf. on Acoustics, Speech, and Signal Processing 1989 (ICASSP '89), pp. 988-991.

The `libjpeg jfdctfst.c` algorithm is based on Arai, Agui, and Nakajima's algorithm for scaled DCT. Their original paper (Trans. IEICE E-71(11):1095) is in Japanese, but the algorithm is described in the Pennebaker and Mitchell's *JPEG Still Image Data Compression Standard* textbook.

The benchmark's input data is a series of .MPEG files, and the output is a series of .PPM files, which can be viewed using any suitable graphic file viewer. Correctness is checked by Cyclical Redundancy Checksum (CRC checking); quality is measured using Peak Signal to Noise Ratio analysis (PSNR). CRC is used as a checkpoint only, not as a canonical validation. Out-of-the-box certifications, most of the time and for most compilers, will have the same CRC values.

The datasets are a superset of the MPEG-2 Encode datasets in terms of the number of frames processed.

Description of Datasets



Graphic

Graphic is a black background ray-traced sequence with reflections, combined with moving light sources with coronas.

The primary elements are the reflections, a secondary halo from the first light source, and a few small artifacts on the front-most graphic.

It is derived from an MPEG transport stream of encapsulated video.

The MPEG 2 parameter file is set to NTSC source parameters.

SEQUENCE MPEG2 MP@ML 720x480 chroma 360x240 fps 30

maxBps 1000000 vbv 229376

Picture 720x480 display 720x480 pixel 8x9

A sequence of 50 frames is used for encoding. This results in a 3 second run, and keeps the RAM file requirements under 4 megabytes. The resulting MPEG file size is 232,677 bytes.

Railgrind

Railgrind is a sequence of a skateboarder doing a grind move down a handrail and landing in an open space. The camera is centered on the skateboarder, which results in a fast moving color background.

The artifacts to watch for are tearing of the lower background at the bottom part of the rail move.

The original file is an MPEG system stream with video on channel 0.

SEQUENCE MPEG2 MP@ML PROG 320x240 chroma 160x120

fps 25 maxBps 100000 vbv 65536

Picture 320x240 display 320x240 pixel 1x1

A sequence of 30 frames is used for railgrind decoding.

Sign

"Sign" shows a person using sign language. There is a zoom-in effect to the speaker, with a complex color background of people.

It is derived from an MPEG system stream with video on channel 1.

Artifacts may appear as small color blocks appearing in the bottom lines of the picture.

The original input dimensions are 352x256, however the decoder only correctly decodes this with image size at 352x240

SEQUENCE MPEG2 MP@ML PROG 352x240 chroma 176x120 fps 25

maxBps 95000 vbv 32768

Picture 352x240 display 352x240 pixel 1x1

300 frames of sign are decoded.

Zoom

Zoom is a beach scene with a rapid zoom-out effect. The original input was an AVI file, extracted to bitmaps. These bitmaps were converted to PPM files, and re-decoded at 30 fps. The final YUV files were generated from the decoded file.

SEQUENCE MPEG2 MP@ML PROG 320x240 chroma 160x120

fps 30 maxBps 95000 vbv 65536

Picture 320x240 display 320x240 pixel 1x1

65 frames are used for decoding.

Marsface

Marsface is a rotating black and white radar picture of a Mars feature. The feature is 3 dimensional with a perspective view. The original is a 24-fps MPEG file. This file format is maintained for the decoder. For encoding, the bitrate is increased as well as the fps. An fps of 25 is the closest available setting in the decoder.

Original Attributes:

SEQUENCE PROG 192x192 chroma 96x96 fps 24

maxBps 0 vbv 32768

picture 192x192 display 192x192 pixel 1x1

Generated attributes:

SEQUENCE MPEG2 MP@ML PROG 192x192 chroma 96x96

fps 25 maxBps 95000 vbv 65536

picture 192x192 display 192x192 pixel 1x1

All 49 frames are used for decoding.

Benchmark Processing

Processing consists of:

7. Read the MPEG-2 file
8. Read and interpret the header information
9. Read and decode frames of data
10. Process the data based on the header information
11. Output the .PPM file into memory
12. Calculate a PSNR value

A single iteration of the benchmark is complete when the end of the input file is reached and no more data is available to be processed.

Output quality is measured using Peak Signal to Noise Ratio (PSNR) code developed by EEMBC. PSNR is a decibel measurement of noise power and is consistent for the industry, and widely used to measure picture and audio quality. PSNR is measured outside the benchmark timing loop.

Analysis of Computing Resources

This benchmark concentrates mostly on computational processing rather than file I/O, with the key algorithms being the inverse discrete cosine transform.

DENBench™ Version 1.0

Benchmark Name: MPEG-2 Encode

Highlights

- **Five different test files to stress different aspect of encoders**
- **Floating point and integer implementations**
- **Implements PSNR to check output quality**

Application

The MPEG-2 Encode benchmark provides an indication of the potential performance of a microprocessor running an MPEG-2 encoder application.

Benchmark Description

The benchmark contains two different variations: an optional single-precision floating-point algorithm and a fixed-point version derived from ISO sources. MPEG-2 Encode uses a fairly standard reference implementation of the core algorithm, including Huffman decoding and modified inverse discrete cosine transform (iDCT) routines.

The fixed-point implementation base algorithm of `fdctint.c` and `jfdctint.c` is based on Loeffler et al (1989) [1].

For testing purposes, the benchmark was built and tested under Microsoft Windows using gcc and Visual C, Diab Data for PowerPC, gcc under Solaris (64-bit), Green Hills Software for ARM, and other compilers.

The input is a series of five datasets, which take the form of .PPM files along with YUV. The output is an MPEG file (.MPEG) file which can be played using Windows Media Player or Apple Quicktime to help verify that the encoding was correct (assuming you use the uuencode option in the Test Harness). Correctness is also checked by cyclical redundancy checksum (CRC checking), and we measure quality using peak signal-to-noise ratio analysis. CRC is used as a checkpoint only, not as a canonical validation. Out-of-the-box certifications, most of the time, for most compilers, will have the same CRC values.

The Datasets

Dataset #1: "Graphic"

Description

"Graphic" is a black-background ray-traced sequence with reflections and moving light sources with coronas. The primary elements are the reflections, a secondary halo from the first light source, and a few small artifacts on the front-most graphic. It is derived from an MPEG transport stream of encapsulated video.

The MPEG 2 parameter file is set to the following NTSC source parameters:

SEQUENCE MPEG2 MP@ML 720x480 chroma 360x240 fps 30

maxBps 1000000 vbv 229376

Picture 720x480 display 720x480 pixel 8x9

The Encoding Process

A sequence of seven frames is used for encoding. This results in a three-second run and keeps the ram file requirements under 4 Mbytes.

The resulting MPEG file size is 232,677 bytes. On a 1.6-GHz reference platform, one iteration of the encoding process takes 3.57 seconds total run time.

- **Dataset #2: "Railgrind"**

Description

"Railgrind" shows a skateboarder performing a grind move down a handrail and landing in an open space. The camera is centered on the skateboarder, which results in a fast moving color background.

The artifacts to watch for are tearing of the lower background at the bottom part of the rail move.

The dataset uses a 135-decoder source and a 30-frame encoder source. The original is an MPEG system stream with video on channel 0.

SEQUENCE MPEG2 MP@ML PROG 320x240 chroma 160x120

fps 25 maxBps 100000 vbv 65536

Picture 320x240 display 320x240 pixel 1x1

The Encoding Process

A sequence of 30 frames is used for railgrind encoding. The resulting MPEG-2 file size is 122,578 bytes. The total runtime for one iteration of the encoding process is 2.3 seconds on a 1.6-GHz reference platform.

Dataset #3: "Sign"

Description

"Sign" shows a person using sign language. There is a zoom-in effect to the speaker, with a complex color background of people.

It is derived from an MPEG system stream with video on channel 1.

Artifacts may appear as small colorblocks appearing in the bottom lines of the picture.

The dataset uses a 300-frame decoder source and a 30-frame encoder source. The original input dimensions are 352x256, but the encoder only

correctly decodes this with image size set to 352x240.

SEQUENCE MPEG2 MP@ML PROG 352x240 chroma 176x120 fps 25

maxBps 95000 vbv 32768

Picture 352x240 display 352x240 pixel 1x1

The Encoding Process

The first 30 frames of sign are used for encoding. The resulting MPEG-2 file size is 118,940 bytes. The total runtime for one iteration of the encoding process is 1.9 seconds on a 1.6-GHz reference platform.

Dataset #4: "Zoom"

Description

"Zoom" is a beach scene with a rapid zoom-out effect. The original input was an AVI file, extracted to bitmaps. These bitmaps were converted to PPM files and re-encoded at 30 fps. The final YUV files were generated from the encoded file.

"Zoom" uses a 65-frame decoder source and a 30-frame encoder source.

SEQUENCE MPEG2 MP@ML PROG 320x240 chroma 160x120

fps 30 maxBps 95000 vbv 65536

Picture 320x240 display 320x240 pixel 1x1

The Encoding Process

The first 30 frames are used for encoding. The resulting MPEG-2 file size is 96,170 bytes. The total runtime for one iteration of the encoding process is 3.3 seconds on a 1.6-GHz reference platform.

Dataset #5: "Marsface"

Description

Marsface is a rotating black and white radar picture of a Mars feature. The feature is three dimensional with a perspective view.

Marsface uses 49-frame decoder and encoder sources.

The original is a 24-fps MPEG file. This file format is maintained for the decoder. For encoding, the bitrate is increased as well as the fps rate. An fps rate of 25 is the closest available setting in the encoder.

Original Attributes:

SEQUENCE PROG 192x192 chroma 96x96 fps 24

maxBps 0 vbv 32768

picture 192x192 display 192x192 pixel 1x1

Generated attributes:

SEQUENCE MPEG2 MP@ML PROG 192x192 chroma 96x96

fps 25 maxBps 95000 vbv 65536

picture 192x192 display 192x192 pixel 1x1

The Encoding Process

All 49 frames are used for encoding.

The resulting MPEG-2 file size is 70,209 bytes. The total runtime for one iteration of the encoding process is 1.4 seconds on a 1.6-GHz reference platform.

Processing consists of:

13. Reading the selected YUV frames.
14. Reading and interpreting the header information.
15. Reading and encoding frames of data
16. Processing the data based on the header information
17. Outputting the .mpeg file into memory
18. Calculating a PSNR value

A single iteration of the benchmark is complete when the end of the input file is reached and no more data is available to be processed.

Quality Measurements

EEMBC has developed a proprietary methodology for measuring the quality of the MPEG-2 output based on peak signal-to-noise ratio (PSNR) code. PSNR is a decibel measurement of noise power used widely and consistently to measure picture and audio quality. In the EEMBC benchmarks, PSNR is measured outside the benchmark timing loop and on the host, not on the target board.

Double-Ended Signal Quality Measurement

The PSNR methodology implemented by EEMBC is enhanced to provide individual scores for the encode and decode steps. This is still double-ended signal quality measurement; we have just introduced a second set of encoder reference files, and host processing steps, to provide additional information.

The host decoder is used to convert the output files from the embedded encoder. These result files are then used to calculate the PSNR score for the encoder. The final results for PSNR are a fundamentally different calculation from benchmark timing. The Test Harness produces a benchmark timing score represented as a single number.

For PSNR, each benchmark is required to produce large volumes of data, i.e. on the order of several megabytes. This log file data is post-processed on the host to generate a collection of PSNR scores for each benchmark. The PSNR

scores are aggregated by geometric mean. The traditional tab-delimited log file for benchmark timing plus an additional file in comma-separated value (.csv) format is produced for PSNR. Both summary files are readable by spreadsheet programs.

PSNR requires individual frame files from the target and a set of reference files for comparison. The YUV file format involves three separate files for each frame.

PSNR Utility

A utility program called **PSNR.exe** consists of the following components:

1. Math to calculate PSNR of two comparison images, or frames using sum of squared distances method. The advantage of this method is that it can handle images of different dimensions and stride. It is also efficient for handling the volume of files we are processing in a reasonable time.
2. Math to calculate PSNR of two integer arrays used for processing PCM data. The bit depth of the PCM data is also used to support 8-, 16-, 24-, and 32-bit PCM data as required.
3. PSNR aggregation methods which include:
   a. geometric mean calculation
   b. arithmetic mean calculation
   c. sample variance calculation
   d. detection and accumulation of exact match frames (PSNR infinity)
   e. detection and accumulation of all zero frames (PSNR infinity)
4. File processing routines to locate files in the EEMBC tree paths and filename processing to generate frame file names based on file format. This minimizes the post processing steps by locating all of the files required with a few parameters.
5. Image processing to determine file types, the file formats used for reading YUV12 and AIFF formats, and to output YUV12 as PGM and AIFF data as PCM.
6. A standard method for reviewing frames by developers with three types of error images for YUV, and PCM files for MP3. Error images can be generated to identify specific problems during porting. They can also be used in certification when verification beyond basic PSNR is needed.

Knee (2000) [2] is a relevant overview of PSNR, the error image, and its usage to evaluate quality.
(www.broadcastpapers.com/sigdis/Snell&WilcoxQualityMeasure02.htm)

Analysis of Computing Resources

There are two implementations, fixed point and floating point. The floating-point version is optional. This is a benchmark that concentrates mostly on computational processing rather than file I/O. PSNR scores must be reported.

References

[1] Loeffler, C., A. Ligtenberg and G. Moschytz. "Practical Fast 1-D DCT Algorithms with 11 Multiplications," *Proc. Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP 89), pp. 988-991.

[2] Knee, Mike. "A Single-Ended Picture Quality Measure for MPEG-2," (2000); http://www.broadcastpapers.com/sigdis/Snell&WilcoxQualityMeasure02.htm

DENBench™ Version 1.0

Benchmark Name: MPEG-4 Decode

Highlights

- **Benchmarks potential performance of an MPEG-4 decoder**
- **Five different test files stress different encoder aspects**
- **Integer implementation**
- **Based on open source XviD code base**
- **Implements PSNR to check the output quality**

Application

The MPEG-4 decode benchmark provides an indication of the potential performance of a microprocessor subsystem running an MPEG-4 decoder application. MPEG-4 encode and decode are both popular in mobile devices and on the Internet, as well as in digital television and video over IP applications. We implement the XviD codec and code base with modifications for benchmarking and proprietary datasets.

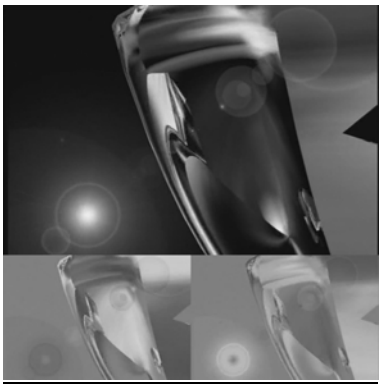
Benchmark Description

EEMBC's XviD implementation uses simple profile/level 3 to encode the files. We are decoding the results of the MPEG-4 Encoder. The input data sets (YUV files) are the same as for EEMBC's MPEG-2 benchmarks, and the output of the MPEG-4 Decode benchmarks are mp4u files that can be compared to their MPEG-2 counterparts.

The benchmark contains a fixed-point integer implementation.

The input is an mp4u (MPEG4 raw container) file, and the output is series of YUV files which can be viewed using GIMP or another picture viewer to help verify that the decoding was correct (assuming you use the `uuencode` option in the Test Harness. To verify correctness of the output we measure the quality using Peak Signal to Noise Ratio analysis.

Description of Datasets



Graphic

Graphic is a black background ray traced sequence with reflections, and moving light sources with coronas.

The primary elements are the reflections, a secondary halo from the first light source, and a few small artifacts on the front most graphic.

It is derived from an mpeg transport stream of encapsulated video.

Rail Grind

1. Notes
    a. 135 frame decoder source
    b. 30 frame encoder source

Rail Grind is a sequence of a skateboarder doing a grind move down a handrail and landing in an open space. The camera is centered on the skateboarder, which results in a fast moving color background.

The artifacts to watch for are tearing of the lower background at the bottom part of the rail move.

The original is an mpeg system stream with video on channel 0.

Sign

1. Notes
    a. 300 frame decoder source
    c. 30 frame encoder source

Sign shows a person using sign language. There is a zoom-in effect to the speaker, with a complex color background of people.

It is derived from an mpeg system stream with video on channel 1.

The artifacts to look for are some small colorblocks appearing in the bottom lines of the picture.

Zoom

1. Notes
    a. 65 frame decoder source
    b. 30 frame encoder source

Zoom is a beach scene with a rapid zoom out effect.

The original input was an AVI file, extracted to bitmaps. These bitmaps were converted to PPM files, and re-encoded at 30 fps. The final YUV files were generated from the encoded file.

Marsface

1. Notes
    a. 49 frame decoder source
    d. 49 frame encoder source

Marsface is a rotating black and white radar picture of a Mars feature. The feature is three dimensional with a perspective view.

Benchmark Processing

Processing consists of:

19. Reading the selected YUV frames.
20. Reading and interpreting the header information.
21. Read and encode frames of data
22. Process the data based on the header information
23. Output the .mpeg file into memory
24. A PSNR value is calculated

A single iteration of the benchmark is complete when the end of the input file is reached and no more data is available to be processed.

Quality Measurements

EEMBC has developed a proprietary methodology for measuring the quality of the MPEG-2 output based on peak signal-to-noise ratio (PSNR) code. PSNR is a decibel measurement of noise power used widely and consistently to measure picture and audio quality. In the EEMBC benchmarks, PSNR is measured outside the benchmark timing loop and on the host, not on the target board.

Double-Ended Signal Quality Measurement

The PSNR methodology implemented by EEMBC is enhanced to provide individual scores for encode and decode steps. This is still double-ended signal quality measurement; we have just introduced a second set of encoder reference files, and host processing steps, to provide additional information.

The host decoder is used to convert the output files from the embedded encoder. These result files are then used to calculate the PSNR score for the encoder. The final results for PSNR are a fundamentally different calculation from benchmark timing. The Test Harness produces a benchmark timing score represented as a single number.

For PSNR, each benchmark is required to produce large volumes of data, i.e. on the order of several megabytes. This log file data is post-processed on the host to generate a collection of PSNR scores for each benchmark. The PSNR scores are aggregated by geometric mean. The traditional tab-delimited log file for benchmark timing plus an additional file in comma-separated value (.csv) format is produced for PSNR. Both summary files are readable by spreadsheet programs.

PSNR requires individual frame files from the target and a set of reference files for comparison. The YUV file format involves three separate files for each frame.

PSNR Utility

A utility program called **PSNR.exe** consists of the following components:

7. Math to calculate PSNR of two comparison images, or frames using sum of squared distances method. The advantage of this method is that it can handle images of different dimensions and stride. It is also efficient for handling the volume of files we are processing in a reasonable time.
8. Math to calculate PSNR of two integer arrays used for processing PCM data. The bit depth of the PCM data is also used to support 8-, 16-, 24-, and 32-bit PCM data as required.
9. PSNR aggregation methods which include:
    a. geometric mean calculation
    b. arithmetic mean calculation
    c. sample variance calculation
    d. detection and accumulation of exact match frames (PSNR infinity)
    e. detection and accumulation of all zero frames (PSNR infinity)
10. File processing routines to locate files in the EEMBC tree paths and filename processing to generate frame file names based on file format. This minimizes the post processing steps by locating all of the files required with a few parameters.
11. Image processing to determine file types, the file formats used for reading YUV12 and AIFF formats, and to output YUV12 as PGM and AIFF data as PCM.
12. A standard method for reviewing frames by developers with three types of error images for YUV, and PCM files for MP3. Error images can be generated to identify specific problems during porting. They can also be used in certification when verification beyond basic PSNR is needed.

Knee (2000) is a relevant overview of PSNR, the error image, and its usage to evaluate quality. (www.broadcastpapers.com/sigdis/Snell&WilcoxQualityMeasure02.htm)

Analysis of Computing Resources

The MPEG-4 Decode benchmark is offered in a fixed-point version only. This is a benchmark that concentrates mostly on computational processing rather than file I/O. PSNR scores must be reported to qualify for certification and publication.

Optimizations Allowed

Out of the Box/Standard C
Full Fury/Optimized

- The C code must not be changed for out of the box unless it must be modified to get it to compile. All

changes must be documented and must not have a performance impact.

- For Out of the Box, additional hardware can be used if it does not require code changes.
- All optimized libraries must be part of the standard compiler package, and/or available to all customers.
- The EEMBC Test Harness Regular or Test Harness Lite may be used. Test harness changes may be made for portability reasons if they do not impact performance.
- For Optimized, the basic algorithm may be changed and/or the code can be rewritten in assembler. We report PSNR scores to help you judge quality of computational processing.
- For Optimized, optimized libraries can be used if they are publicly available.
- For Optimized, hardware-assist can be used if it is on the same processor as that being benchmarked.
- For Optimized, in-lining is allowed.
- Additional data files are used by the EEMBC Technology Center (ETC) during certification to ensure the correctness of the optimized benchmark. You should not assume data patterns during optimization.

DENBench™ Version 1.0

Benchmark Name: MPEG-4 Encode

Highlights

- **Benchmarks potential performance of an MPEG-4 encoder**
- **Five different test files stress different encoder aspects**
- **Integer implementation**
- **Based on open source XviD code base**
- **Implements PSNR to check the output quality**

Application

The MPEG-4 Encode benchmark provides an indication of the potential performance of a microprocessor subsystem running an MPEG-4 encoder application. MPEG-4 encode and decode are both popular in mobile devices and on the Internet, as well as in digital television and video over IP applications. We implement the XviD codec and code base with modifications for benchmarking and proprietary datasets.

Benchmark Description

EEMBC's XviD implementation uses simple profile/level 3 to encode the files. Since we generate the encoded files from raw YUV images, we have the option to create encoded MPEG-4 files with different profiles if needed.

The input data sets (YUV files) are the same as for EEMBC's MPEG-2 benchmarks.

Within the MPEG-4 standard, profiles and levels are defined to ensure interoperability between encoders and decoders. Profiles restrict the MPEG-4 features used, such as b-frames and quarterpel interpolation, while levels impose restriction on bit rate, memory, and complexity. These profiles are published in the MPEG-4 visual standard and available informally from the M4IF website. DivXNetworks has also defined its own profiles and levels, which are supported by hardware carrying the DivX Certified logo. XviD, the open source code base, is DivX spelled backwards.

The XviD encoder can be configured to generate any of the profiles above, although often it is configured to generate content using the complete ASP feature set. In addition, XviD presently does not perform video-buffer-verification, and as such, XviD content may not conform to the bit rates specified in any levels (ISO/IEC, DivX, or otherwise).

The M4IF definitions and discussion of profile and level are here:

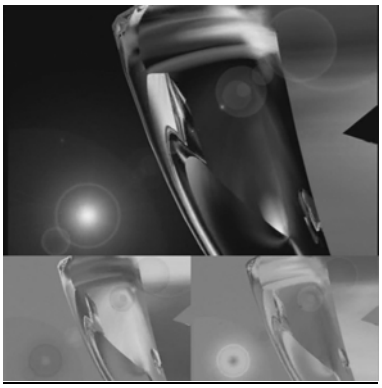
http://www.m4if.org/resources/profiles/index.php

The benchmark contains two different variations: a fixed-point integer and an optional single-precision floating point (f_32) implementation.

The input is an a series of .PPM files along with YUV, and the outputs are mp4u files which can be decoded by XVid to help verify that the encoding was correct (assuming you use uuencode option in the Test Harness). Correctness test is also performed by measuring quality using Peak Signal to Noise Ratio analysis.

Description of Datasets

Graphic



Graphic is a black background ray traced sequence with reflections, and moving light sources with coronas.

The primary elements are the reflections, a secondary halo from the first light source, and a few small artifacts on the front most graphic.

It is derived from an mpeg transport stream of encapsulated video.

Graphic MPEG-4 Encode

A sequence of 7 frames is used for encoding. This results in a 3-second run, and keeps the ram file requirements under 4 Mb.

Rail Grind

2. Notes
    a. 135 frame decoder source
    b. 30 frame encoder source

Rail Grind is a sequence of a skateboarder doing a grind move down a handrail and landing in an open space. The camera is centered on the skateboarder, which results in a fast moving color background.

The artifacts to watch for are tearing of the lower background at the bottom part of the rail move.
The original is an mpeg system stream with video on channel 0.

Rail Grind MPEG-4 Encoding

A sequence of 30 frames is used for railgrind encoding.

Sign

2. Notes
    a. 300 frame decoder source
    c. 30 frame encoder source

Sign shows a person using sign language. There is a zoom-in effect to the speaker, with a complex color background of people.

It is derived from an mpeg system stream with video on channel 1.

The artifacts to look for are some small colorblocks appearing in the bottom lines of the picture.

The original input dimensions are 352x256, however the encoder only correctly decodes this with image size set to 352x240.

Sign MPEG-4 Encoding
The first 30 frames of sign are used for encoding.

Zoom

2. Notes
    a. 65 frame decoder source
    b. 30 frame encoder source

Zoom is a beach scene with a rapid zoom out effect.

The original input was an AVI file, extracted to bitmaps. These bitmaps were converted to PPM files, and re-encoded at 30 fps. The final YUV files were generated from the encoded file.

Zoom MPEG-4 Encoding
The first 30 frames are used for encoding.

Marsface

2. Notes
    a. 49 frame decoder source
    d. 49 frame encoder source

Marsface is a rotating black and white radar picture of a Mars feature. The feature is three dimensional with a perspective view.

The original is a 24-fps mpeg file. This file format is kept for the deocder. For encoding, the bitrate is increased as well as the fps. An fps of 25 is the closest available setting in the encoder.

Original Attributes:
SEQUENCE PROG 192x192 chroma 96x96 fps 24 maxBps 0 vbv 32768
picture 192x192 display 192x192 pixel 1x1

Marsface MPEG-4 Encoding
All 49 frames are used for encoding.

Benchmark Processing

Processing consists of:

25. Reading the selected YUV frames.
26. Reading and interpreting the header information.
27. Read and encode frames of data
28. Process the data based on the header information
29. Output the .mpeg file into memory
30. A PSNR value is calculated

A single iteration of the benchmark is complete when the end of the input file is reached and no more data is available to be processed.

Quality Measurements

EEMBC has developed a proprietary methodology for measuring the quality of the MPEG-2 output based on peak signal-to-noise ratio (PSNR) code. PSNR is a decibel measurement of noise power used widely and consistently to measure picture and audio quality. In the EEMBC benchmarks, PSNR is measured outside the benchmark timing loop and on the host, not on the target board.

Double-Ended Signal Quality Measurement

The PSNR methodology implemented by EEMBC is enhanced to provide individual scores for encode and decode steps. This is still double-ended signal quality measurement; we have just introduced a second set of encoder reference files, and host processing steps, to provide additional information.

The host decoder is used to convert the output files from the embedded encoder. These result files are then used to calculate the PSNR score for the encoder. The final results for PSNR are a fundamentally different calculation from benchmark timing. The Test Harness produces a benchmark timing score represented as a single number.

For PSNR, each benchmark is required to produce large volumes of data, i.e. on the order of several megabytes. This log file data is post-processed on the host to generate a collection of PSNR scores for each benchmark. The PSNR scores are aggregated by geometric mean. The traditional tab-delimited log file for benchmark timing plus an additional file in comma-separated value (.csv) format is produced for PSNR. Both summary files are readable by spreadsheet programs.

PSNR requires individual frame files from the target and a set of reference files for comparison. The YUV file format involves three separate files for each frame.

PSNR Utility

A utility program called **PSNR.exe** consists of the following components:

13. Math to calculate PSNR of two comparison images, or frames using sum of squared distances method. The advantage of this method is that it can handle images of different dimensions and stride. It is also efficient for handling the volume of files we are processing in a reasonable time.
14. Math to calculate PSNR of two integer arrays used for processing PCM data. The bit depth of the PCM data is also used to support 8-, 16-, 24-, and 32-bit PCM data as required.
15. PSNR aggregation methods which include:
    a. geometric mean calculation
    b. arithmetic mean calculation
    c. sample variance calculation
    d. detection and accumulation of exact match frames (PSNR infinity)
    e. detection and accumulation of all zero frames (PSNR infinity)
16. File processing routines to locate files in the EEMBC tree paths and filename processing to generate frame file names based on file format. This minimizes the post processing steps by locating all of the files required with a few parameters.
17. Image processing to determine file types, the file formats used for reading YUV12 and AIFF formats, and to output YUV12 as PGM and AIFF data as PCM.
18. A standard method for reviewing frames by developers with three types of error images for YUV, and PCM files for MP3. Error images can be generated to identify specific problems during porting. They can also be used in certification when verification beyond basic PSNR is needed.

Knee (2000) is a relevant overview of PSNR, the error image, and its usage to evaluate quality. (www.broadcastpapers.com/sigdis/Snell&WilcoxQualityMeasure02.htm)

Analysis of Computing Resources

There are two implementations, fixed point and floating point. The floating-point version is optional. This is a benchmark that concentrates mostly on computational processing, rather than file I/O. PSNR scores must be reported.

Optimizations Allowed

Out of the Box/Standard C
Full Fury/Optimized

- The C code must not be changed for out of the box unless it must be modified to get it to compile. All

changes must be documented and must not have a performance impact.

- For Out of the Box, additional hardware can be used if it does not require code changes.
- All optimized libraries must be part of the standard compiler package, and/or available to all customers.
- The EEMBC Test Harness Regular or Test Harness Lite may be used. Test harness changes may be made for portability reasons if they do not impact performance.
- For Optimized, the basic algorithm may be changed and/or the code can be rewritten in assembler. We report PSNR scores to help you judge quality of computational processing.
- For Optimized, optimized libraries can be used if they are publicly available.
- For Optimized, hardware-assist can be used if it is on the same processor as that being benchmarked.
- For Optimized, in-lining is allowed.
- Additional data files are used by the EEMBC Technology Center (ETC) during certification to ensure the correctness of the optimized benchmark. You should not assume data patterns during optimization.

DENBench™ Version 1.0

Benchmark Name: RGB to CMYK Conversion

Highlights

- **Benchmarks digital image processing performance in printers and other digital imaging products**
- **Explores basic arithmetic and minimum value detection capability**
- **Provides opportunities for Full Fury benchmark optimization**
- **Conditional move and multi-byte processing, exercising SIMD and VLIW architectures**
- **Integer implementation**
- **Seven datasets provide a larger workload compared to the single dataset of ConsumerBench Version 1.1**
- **Input is comprised of .ppm files**
- **Implements Non-Intrusive Cyclical Redundancy Checksum (CRC) to Check Output Quality**

Application

RGB to CMYK conversion is widely used in color printers. RGB inputs from PC data are converted to CMYK color signals for printing.

Benchmark Description

This benchmark explores the target CPU's ability to perform basic arithmetic and minimum value detection. The R, G, B 8-bit pixel color image input is fed to the following equation:

```
/* calculate complementary colors */
c = 255 – R;
m = 255 – G;
y = 255 – B;
/* find the black level k */
K = minimum (c,m,y)
/* correct complementary color lever based on k
*/
C = c – K
M = m – K
Y = y - K
```

RGB values are in the range of [0:255]

CMYK values are in the range of [0:255]

The input and output data sizes vary. For example, the 320x240 data for RGB and CMYK is stored sequentially as:

R[0], G[0], B[0], R[1], G[1], B[1], R[76799], G[76799], B[76799]

C[0], M[0], Y[0], K[0], C[1], M[1], Y[1],K[1] C[76799], M[76799], Y[76799], K[76799]

The pointers are incremented by one to access R, G, B or C, M, Y, K data in this order. If the benchmark score is extrapolated for a larger image, the processing time will be almost linearly proportional to the pixel count (e.g. for a 640 x 480 image, it will be multiplied times 4). The iteration/second score will be the inverse (e.g. for a 640 x 480 image, iterations/sec will be multiplied by .25).

There is data dependency in the cycle counts for the minimum value K search, due to branch taken or not taken. If this operation is handled by conditional move, the cycle will be constant.

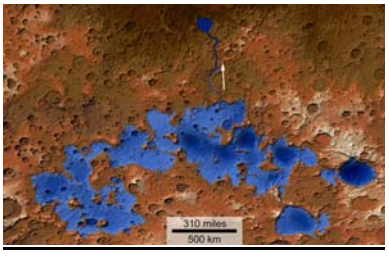
Description of Datasets

Rose Small

Rose Small is the default file for the JPEG Compression benchmark. It is a single image that is contained in both BMP and JPEG formats. The dimensions are 227x149, 256 colors. The image contains 256 unique colors.

Goose

Goose is the default file for the JPEG Decompression benchmark. It is a single image that is contained in both BMP and JPEG formats. The dimensions are 320x240, 256 colors. The image has 22,921 unique colors.



Mars Former Lakes

Mars Former Lakes is a NASA graphics picture. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 800x482, 16 million colors. The image has 91,152 unique colors.

Dragon Fly

Dragon Fly is an image containing highlights, and a wide range of contrast. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 606x896, 16 million colors. The image has 162,331 unique colors.

EEMBC Group Shot

EEMBC Group Shot is a snapshot of EEMBC Board of Directors members at a 2003 meeting. It has a large number of flesh tones, and the highest number of unique colors in the library. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 640x480, 16 million colors. The image has 181,872 unique colors.

David and Dogs

David and Dogs is a snapshot of David Weiss and his dogs Sandy, Toga, and Trudy during a rare snowstorm in Austin. It is used as a grayscale image, with good contrast details in the melting snow. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 564x230, 256 shades of gray. The image has 215 unique colors.

Mandrake

Mandrake is a close up picture of a **Mandrill Baboon** (sometimes misnamed as "Mandrake"). It has a lot of detail and colors. It has been the default image for the filter benchmarks in both color and gray scale. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 320x240, 16 million colors. The image has 71,482 unique colors.

Galileo

Galileo is a NASA composite image based on actual images of the Jupiter and several of its moons. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 290x415, 16 million colors. The image has 36,557 unique colors, and also contains "real black" for over 30% of the picture, which is interesting from an optimization perspective.

Output quality is measured using Non Intrusive CRC code developed by the EEMBC Certification Laboratory (ECL, LLC). It does not affect the benchmark score.

Analysis of Computing Resources

A "for loop" calculates the conversion of one set of RGB inputs and CMYK outputs at a time. A set of R, G, B input data is read from the memory by incrementing a read pointer. A set of C, M, Y, K output data is written back to the memory by incrementing a write pointer. There is no complex two-dimensonal access.

The complementary color calculation and correction are simple subtract calculations without any MAC operation.

The minimum value search has two branches for processing each pixel.

```
If (c<m) {
K = (Byte)(c<y ? c:y);
}
else {
K = (Byte)(m<y ? m:y);
}
```

This can be a very expensive routine because of the branch penalty.

Full-Fury Optimization: By using the compare and conditional moves, the branch penalty can be avoided. VLIW and SIMD can process multiple bytes of data at a time. For example, a four-way SIMD microprocessor can handle 4 x 8-bit data every cycle.

DENBench™ Version 1.0

Benchmark Name: RGB to YIQ Conversion

Highlights

- **Benchmarks digital video processing performance**
- **Provides opportunities for Full Fury optimization**
- **Integer implementation**
- **Seven datasets expand workload compared to comparable benchmark in ConsumerBench Version 1.1**
- **Input is comprised of .ppm files**
- **Implements Non-Intrusive Cyclical Redundancy Checksum (CRC) to Check Output Quality**

Application

RGB to YIQ conversion is used in the NTSC encoder where the RGB inputs from the camera are converted to luminance (Y) and chrominance (I,Q) information. In the NTSC encoder, the I,Q signals are modulated by a subcarrier and added to the Y signal. Historically, when color TVs appeared in the market, they had to coexist with the existing monochrome TVs and this was made possible with the NTSC signal structure. The chrominance signals are averaged out as a fine mesh of invisible signals in the monochrome TV sets. YUV used in the PAL standard and YCbCr used in the JPEG standard have different encodings. All three standards share the same luminance signal Y but the chrominance calculations are different. The matrix calculation scheme used in the RGB to YIQ can be applied to these standards too.

In the actual products, this trivial calculation is usually performed in dedicated hardware, especially in digital video products. For cost saving and flexibility, this algorithm can be implemented in software if the CPU is powerful enough and where the digital image is a still picture.

Benchmark Description

This benchmark explores the capability of the CPU to perform a straightforward matrix multiply/accumulate calculation.

The R, G, B 8-bit pixel color image input is processed as follows:

```
Y = 0.299*R + 0.587*G + 0.114*B

I = 0.596*R – 0.275*G – 0.321*B
```

```
Q = 0.212*R – 0.523*G + 0.311*B
```

RGB values are in the range of [0:255]. The conversion coefficients are 16 bits. The multiply/accumulate results are shifted right by 16 bits. Before the shift, 1 is added to a bit location right to the LSB of the shifted result for rounding to the nearest integer. The output is 8-bit data. Y is in the range of [0,255] and I,Q in the range of [-127, 127]. The input and output data size is 320 pixels in the horizontal direction and 240 pixels in the vertical direction.

The 320x240 data for RGB and YIQ are stored sequentially as:

R[0], G[0], B[0], R[1], G[1], B[1], R[76799], G[76799], B[76799]

Y[0], I[0], Q[0], Y[1], I[1], Q[1], Y[76799], I[76799], Q[76799]

The pointers are just incremented by one to access R, G,B or Y, I, Q data in this order.

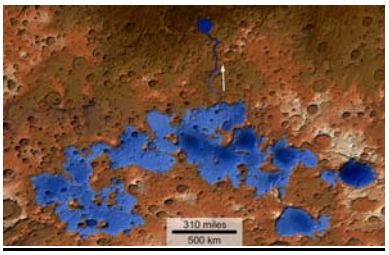
Description of Datasets

Rose Small

Rose Small is the default file for the JPEG Compression benchmark. It is a single image that is contained in both BMP and JPEG formats. The dimensions are 227x149, 256 colors. The image contains 256 unique colors.

Goose

Goose is the default file for the JPEG Decompression benchmark. It is a single image that is contained in both BMP and JPEG formats. The dimensions are 320x240, 256 colors. The image has 22,921 unique colors.



Mars Former Lakes

Mars Former Lakes is a NASA graphics picture. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 800x482, 16 million colors. The image has 91,152 unique colors.

Dragon Fly

Dragon Fly is an image containing highlights, and a wide range of contrast. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 606x896, 16 million colors. The image has 162,331 unique colors.

EEMBC Group Shot

EEMBC Group Shot is a snapshot of EEMBC Board of Directors members at a 2003 meeting. It has a large number of flesh tones, and the highest number of unique colors in the library. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 640x480, 16 million colors. The image has 181,872 unique colors.

David and Dogs

David and Dogs is a snapshot of David Weiss and his dogs Sandy, Toga, and Trudy during a rare snowstorm in Austin. It is used as a grayscale image, with good contrast details in the melting snow. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 564x230, 256 shades of gray. The image has 215 unique colors.

Mandrake

Mandrake is a close up picture of a **Mandrill Baboon** (sometimes misnamed as "Mandrake"). It has a lot of detail and colors. It has been the default image for the filter benchmarks in both color and gray scale. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 320x240, 16 million colors. The image has 71,482 unique colors.

Galileo

Galileo is a NASA composite image based on actual images of the Jupiter and several of its moons. It is a single image that is contained in BMP, PPM, PGM, and JPEG formats. The dimensions are 290x415, 16 million colors. The image has 36,557 unique colors, and also contains “real black” for over 30% of the picture, which is interesting from an optimization perspective.

Output quality is measured using Non Intrusive CRC code developed by the EEMBC Certification Laboratory (ECL, LLC). It does not affect the benchmark score.

Analysis of Computing Resources

Out of the Box Benchmark: A "for loop" calculates the conversion of one set of RGB inputs and YIQ outputs at a time. A set of R, G, B input data is read from the memory by incrementing a read pointer. A set of output Y, I, Q output data is written back to the memory by incrementing a write pointer. There is no complex two-dimensonal access such as that in the high pass grey-scale filter benchmark. The calculation is a straightforward multiplication and accumulation that a microprocessor with a single-cycle MAC unit will benefit from. The code size is small and easily fits in to a small L1 Instruction Cache.

Full-Fury Benchmark: Because of the simple structure of the multiplication and accumulation, a VLIW or SIMD architecture with multiple of MAC units can be used to accelerate performance. A further optimization is the loading of multiple bytes at a time. Software pipelining could be used to pass the loaded data efficiently to the MAC unit for calculation.

DENBench™ Version 1.0

Benchmark Name: RSA

Highlights

- **Benchmarks the Rivest, Shamir, and Adleman (RSA) cryptography algorithm**
- **Created in part from SSLEAY (the open-source Netscape Secure Socket Layer source code base courtesy of Eric Young)**
- **Roundtrip implementation and self-checking assures accuracy**
- **A component of the DENBench cryptography sub-suite**
- **Computationally intensive and accurate implementation of RSA algorithm modified to PKCS standards**
- **Uses Optimal Asymmetric Encryption Padding (OAEP)**

Applications and Restrictions

The RSA algorithm was first described in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT. The letters RSA are the initials of their surnames. According to Wikipedia, RSA was one of the first "strong encryption" public key cryptography schemes. It can be used for both digital signatures and encryption. The RSA cipher is used in numerous cryptographic protocols, including Transport Layer Security (TLS), Secure Socket Layer, (SSL), Secure Shell (SSH), and Internet Protocol Security (IPSEC).

RSA is much slower and therefore more computationally intensive than DES, and unlike DES is not symmetrical. Thus, there are different keys for encryption and decryption.

Although it has been proven to be vulnerable to certain attacks (including timing, man-in-the-middle, and adaptive chosen cipher attacks), it is an extremely popular algorithm used in many e-commerce (internet) and m-commerce (mobile) applications. Some people choose to implement DES, Triple-DES, or AES for stronger encryption. The major concern is really the "shared secret key" nature of the asymmetric system. Because RSA is part of the Secure Socket Layer system used so widely on the internet, and because it can be hacked by determined foes, it is now often paired with Optimal Asymmetric Encryption Padding (hence the term RSA-OAEP), and in fact EEMBC has implemented the benchmark as an RSA-OAEP system. In the benchmark, RSA-OAEP are used together with the Public Key Cryptography Standards (PKCS). The EEMBC code is based on PKCS 1.5 and OAEP 2.0R1 and implements Shoup's improvements to OAEP (in other words, EME-OAEP).

The EEMBC RSA benchmark is a cipher algorithm that provides an indication of the potential performance of a microprocessor or digital signal processor subsystem doing RSA cryptographic encryptions and decryptions.

This benchmark, and the source code, is subject to the following restrictions:

Applications and Restrictions

This software is subject to the following Export Restrictions (exportation from the United States of America to non-USA countries): *Implementations of cryptography are subject to United States Federal Government export controls. Export controls on commercial encryption products are administered by the Bureau of Export Administration (BXA) http://www.bxa.doc.gov/Encryption/ in the U.S. Department of Commerce. Regulations governing exports of encryption are found in the Export Administration Regulations (EAR), 15 C.F.R. Parts 730-774. Compliance with export restrictions is the responsibility of each individual EEMBC member, not EEMBC.*

Benchmark Description

The EEMBC RSA benchmark handling of private key operations does not depend on the private key components being present (for example, a key stored in external hardware). The recommended number of iterations is 30, and it takes about a second to run on a desktop x86 PC at about 1.7 GHz. Checking is by Cyclical Redundancy Checksum (CRC).

Analysis of Computing Resources

The benchmark is computationally challenging: addition, multiplication, extensive use of division, bit shifting, matrix math, bitwise operators such as `XOR`, and other operators are used. It is implemented in integer math. This benchmark is almost exclusively CPU bound, and the quality of the math library as well as memory library has an effect on performance. Memory moves are performed repeatedly, so optimized C library `mem*` functions would improve performance. Use of `malloc()` and heap is extensive, so optimizing memory management will yield better results. Sophisticated superscalar architectures scheduled by sophisticated compilers (or assembly language implementations) can take advantage of some parallelism. Architectures that require aligning for good performance but that do not automatically pad to obtain alignment will suffer. Odd C syntax with numerous breaks and jumps means this benchmark is unlikely to be optimized away by compiler trickery, although good standard optimization techniques (including loop unrolling and hoisting loads) would improve performance. A tool chain must implement a fair fraction of the standard C library, including `rand()` functionality.